1. (Original) A method for compiling Unified Parallel C-language (UPC) source code containing UPC-unique constructs and C-language constructs, the method comprising the steps of:

translating said UPC source code into a first intermediate form;

generating proxy form C-language code strings of data components within said intermediate form UPC-unique constructs;

inserting said generated code strings into said UPC source code to form a combined code;

translating said combined code into a second intermediate form, wherein any statements within said UPC-unique constructs are placed in a C-form with associated program text, and surviving UPC-unique constructs are discarded; and

converting said second intermediate form to compiled machine code.

2. (Original) The method of claim 1, wherein said code strings are proxy declarations.

3. (Currently Amended) The method of claim 2, wherein ~~a~~ each said proxy declaration includes a name that is a mangled version of ~~the~~ a name of ~~the~~ a respective UPC-unique data component having a one-to-one mapping.

4. (Original) The method of claim 1, wherein said associated program text includes a conditional statement.

5. (Currently Amended) The method of claim 4, wherein said UPC-unique ~~statements~~ constructs are forall statements, and said associated program text includes a conditional statement whose predicates leads to evaluation based upon an affinity test.

6. (Currently Amended) The method of claim 5, wherein, for forall statements having an affininty other than ~~"continue"~~ continue, the translation step includes sub-traversal of a forall body and determining the context of each static level of nesting.

7. (Original) The method of claim 6, further comprising the step of incrementing a depth variable in accordance with each said sub-traversal.

8. (Original) A method for compiling UPC source code, comprising the steps of:

(a) converting UPC-unique constructs into C-level form;

(b) inserting said C-level constructs into said source code to form a combined code;

(c) translating said combined code to an intermediate form, wherein any surviving UPC-unique components are discarded; and

(d) converting said intermediate form to compiled machine code.

9. (Original) The method of claim 8, wherein said C-level form constructs are in a form having proxy declarations.

10. (Currently Amended) The method of claim 9, wherein ~~a~~ each said proxy declaration for a UPC-unique data construct includes a name that is a mangled version of ~~the~~ a name of the respective UPC-unique data component having a one-to-one mapping.

11. (Currently Amended) The method of claim 9, wherein said proxy ~~delcaration~~ declaration for a UPC-unique statement includes a conditional statement.

12. (Original) The method of claim 11, wherein, for forall statements, said conditional statement has predicates leading to evaluation based upon an affinity test.

13. (Original) A method for compiling Unified Parallel C-language (UPC) source code containing UPC-unique constructs and C-language constructs, the method comprising the steps of:

translating said UPC source code into a first intermediate form, including any UPC-unique statements being placed in a C-form with associated program text;

generating proxy form C-language code strings of UPC-unique data components within said first intermediate form;

inserting said generated code strings into said UPC source code to form a combined code;

translating said combined code and said C-form statements and associated program text to a second intermediate form, wherein surviving UPC-unique components are discarded; and

converting said second intermediate form to compiled machine code.

14. (Original) The method of claim 13, wherein said code strings are proxy declarations.

15. (Currently Amended) The method of claim 14, wherein ~~a~~ each said proxy declaration includes a name that is a mangled version of ~~the~~ a name of ~~the~~ a respective UPC-unique data component having a one-to-one mapping.

16. (Original) The method of claim 13, wherein said associated program text includes a conditional statement.

17. (Currently Amended) The method of claim 16, wherein said UPC-unique ~~statements~~ constructs are forall statements, and said associated program text includes a conditional statement whose predicates leads to evaluation based upon an affinity test.

18. (Currently Amended) The method of claim 17, wherein, for forall statements having an affininty other than ~~"continue"~~ continue, the translation step includes sub-traversal of a forall body and determining the context of each static level of nesting.

19. (Original) The method of claim 18, further comprising the step of incrementing a depth variable in accordance with each said sub-traversal.

20. (Currently Amended) A UPC compiler comprising:

a front end module operable for receiving UPC source code;

an intermediate form processor operable for converting UPC-unique constructs into C-level form;

a feedback loop to said front end processor,

~~and~~ wherein said front end processor further inserts said C-level constructs into said source code to form a combined code;

~~and further~~ wherein said intermediate form processor translates said combined code to an intermediate form, wherein any surviving UPC-unique components are discarded; and

a back end module <u>operable for</u> converting said intermediate form to compiled machine code.

21. (Original) The compiler of claim 20, wherein said C-level form constructs are in a form having proxy declarations.

22. (Currently Amended) The compiler of claim 21, wherein ~~a~~ <u>each</u> said proxy declaration for a UPC-unique data construct includes a name that is a mangled version of ~~the~~ <u>a</u> name of ~~the~~ <u>a</u> respective UPC-unique data component having a one-to-one mapping.

23. (Original) The compiler of claim 21, wherein said proxy ~~declaration~~ <u>declaration</u> for a UPC-unique statement includes a conditional statement.

24. (Original) The compiler of claim 23, wherein, for forall statements, said conditional statement has predicates leading to evaluation based upon an affinity test.

25. (Currently Amended) The compiler of claim 24, wherein, for forall statements having an affininty other than ~~"continue"~~ <u>continue</u>, the intermediate form procesor translatates by sub-traversal of a forall body and determination of the context of each static level of nesting.

26. (Original) The compiler of claim 25, wherein the intermediate form processor increments a depth variable in accordance with each said sub-traversal.

27. (Cancelled).

28. (Cancelled).

Please add the following new claims:

29. (New) A program storage device readable by computer, tangibly embodying a program of instructions executable by said computer to perform a method for compiling UPC source code, said method comprising:

converting UPC-unique constructs into C-level form;

inserting said C-level constructs into said source code to form a combined code;

translating said combined code to an intermediate form, wherein any surviving UPC-unique components are discarded; and

converting said intermediate form to compiled machine code.

30. (New) The program storage device of claim 29, wherein said C-level form constructs are in a form having proxy declarations.

31. (New) The program storage device of claim 30, wherein each said proxy declaration for a UPC-unique data construct includes a name that is a mangled version of a name of the respective UPC-unique data component having a one-to-one mapping.

32. (New) The program storage device of claim 30, wherein said proxy declaration for a UPC-unique statement includes a conditional statement.

33. (New) The program storage device of claim 32, wherein, for forall statements, said conditional statement has predicates leading to evaluation based upon an affinity test.

34. (New) A parallel distributed shared memory computer system having a single real address space, comprising:

a plurality of processor modules;

a memory unit associated with each processor module; and

an interconnection network linking all processor modules and memory units;

and wherein each processor module includes a UPC compiler module, each said UPC compiler module including:

a front end module operable for receiving UPC source code;

an intermediate form processor operable for converting UPC-unique constructs into C-level form;

a feedback loop to said front end processor,

wherein said front end processor further inserts said C-level constructs into said source code to form a combined code;

wherein said intermediate form processor translates said combined code to an intermediate form, wherein any surviving UPC-unique components are discarded; and

a back end module operable for converting said intermediate form to compiled machine code.

35. (New) The computer system of claim 34, wherein said C-level form constructs are in a form having proxy declarations.

36. (New) The computer system of claim 35, wherein each said proxy declaration for a UPC-unique data construct includes a name that is a mangled version of a name of the respective UPC-unique data component having a one-to-one mapping.

37. (New) The computer system of claim 35, wherein said proxy declaration for a UPC-unique statement includes a conditional statement.

38. (New) The computer system of claim 37, wherein, for forall statements, said conditional statement has predicates leading to evaluation based upon an affinity test.